Design and Implementation of Architecture for Multi-Robot Cooperation in the Context of WSN

Milan Erdelj
milan.erdelj@inria.fr
Inria Lille – Nord Europe
France

Tahiry Razafindralambo
tahiry.razafindralambo@inria.fr
Inria Lille – Nord Europe
France

ABSTRACT

The concept of autonomous mobile agents gets a lot of attention in the domain of wireless sensor networks (WSN) or wireless sensor and actuator networks (WSAN). Multiple robots that coordinate or cooperate with other sensors, robots or human operator, allow the WSN/WSAN to perform tasks that are far beyond the scope of single robot unit. In this work, we describe the robot middleware architecture that allows networked multi-robot control and data acquisition in the context of wireless sensor networks. Furthermore, we present three examples of robot network deployment and illustrate the proposed architecture usability: the robotic network deployment with the goal of covering the Point of Interest, adaptable multi-hop video transmission scenario, and the case of obtaining the energy consumption during the deployment.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication;
I.2.9 [**Robotics**]: Sensors

Keywords

communication; implementation; multi-robot; WSN

1. INTRODUCTION

The advances in mobile robotics allow us today to add the mobility concept into many different classes of wireless sensor networks (WSN) or wireless sensor and actuator networks (WSAN) applications. The deployment of mobile sensors is possible and useful in many application scenarios, ranging from the environmental monitoring, e.g., volcano activity, dispersion of fire, pollutants or gas plumes, and public safety applications (event or object surveillance), to the industry (structure and machinery health) and military applications (automated warfare, land mine detection). Manual sensor deployment represents a rather difficult task to achieve in such type of applications due to various reasons. The use of mobile agents, i.e., robotic platforms equipped with sensory and motion capabilities, allows us to overcome these difficulties by deploying the sensor network in a random manner and applying the self-repositioning of self-deploying techniques.

Multiple robots that coordinate or cooperate with other sensors, robots or human operator, allow the WSAN to perform tasks that are far beyond the scope of single robot unit. Indeed, the performance of networks composed of coordinated robots that cooperate in achieving their goal, by far exceeds the performance of one specialized robot that needs to achieve all the tasks given. The improved performance is due to the parallelism introduces in the network. Improved performance induces the improved efficiency, that reflects in the quality of fulfilled goal and the speed of its execution. One of the challenged in networked robotics is the robot intelligence, i.e., the ability recognizing the environment and specific situation, solve the unexpected problems and dynamically reconfigure the network in the case of sudden errors and robot failures.

In this paper, we present robot middleware architecture dedicated to the multi-robot deployment applications. Due to the unpredictability of the deployment environment, followed by the uncertainties regarding the robot robustness, the deployment algorithm should fulfill some basic requirements. It should be *localized* (each robot relies only on the locally available information), *distributed* (each robot performs its calculations concerning neighborhood discovery and path planning), *efficient* (minimal number of robots should be used to achieve the maximum goals as possible by using the minimal amount of resources), *self-reconfigurable* (the network infrastructure should be automatically recoverable in the case of robot failures).

The middleware that we describe in this paper allows the user to easily implement different types of deployment algorithms. We provide the central base-station application that allows a user to gather all the information sensed by the fleet of robots, as well as to send commands to a group or an individual robots, thus introducing the manual control in the robotic network if necessary. The middleware presented in this work is dedicated to be used with Wifibot mobile robots [20]. However, it can be easily adjusted in order to be used with other robotic platforms. We present three examples of robot network deployment and illustrate the proposed architecture usability: the robotic network deployment with the goal of covering the Point of Interest, adaptable multi-hop

PE-WASUN'13, November 3–8, 2013, Barcelona, Spain.

ACM 978-1-4503-2360-4/13/11...$15.00
http://dx.doi.org/10.1145/2507248.2507260.

video transmission scenario, and the case of obtaining the energy consumption during the deployment.

The remainder of the paper is organized as follows: the motivation for this work and the problems of networked robots in the context of WSN are discussed in Section 2. Some related works are analyzed in Section 3. Robot middleware architecture and multi-robot network control details are provided in Section 4. Three WSN applications where our proposed control architecture is used are presented in Section 5. Future works are announced and conclusions are drawn in Section 6.

2. MOTIVATIONS

The motivation for this work is the increased use of the mobile robots in the WSN applications. In order to increase the effectiveness and efficiency of data collection, mobile robots should cooperate in the terms of group movement and the data acquisition. Coordinated mobility plays an important role in every aspect of the WSN applications, such as structural health inspection, environmental monitoring, area surveillance or broken network infrastructure restoration. Mobile robots can be used in order to extend the overall network lifetime by interacting with sensor devices, as well as providing the mobility to the sensors thus increasing the quality of service.

Generally speaking, there are two classes of applications in which the use of autonomous mobile robots is needed:

- inaccessible or unknown deployment environment inspection (warfare field, structural health monitoring, and machinery inspection [13]),
- and the hazardous environment where the presence of humans can be endangered (minefields, toxic gas leaks, and pollutions source detection [9]).

Regarding the first class of applications, the mobile multi-robotic networks play an important role in the field of electronic and visual reconnaissance, deployment field surveillance, target detection, and identification. Most of these applications have a military connotation up to a certain degree, which is understandable since the huge amount of resources are allocated in order to improve the quality and usability of mobile robotic networks. Military applications focusing on security are present in the second class of applications as well, with the applications such as minefield exploration and mine detection, together with chemical, biological, radiological, nuclear, and explosive reconnaissance problems that need to be solved with the help of mobile robotic networks.

One typical example of the robot deployment for such means is presented in [8]. A set of autonomous mobile robotic vehicles is deployed in the building that suffered an attack and therefore represented an unknown environment with unknown number and placement of people inside. The goal given to the set of robots was to explore the ruined building, locate the people inside and to provide the rescuers with the exact information regarding the situation in the building. Needless to say that the speed of the deployment was of the essence and that all the robots had to collaborate in order to save human lives. This example shows that it is possible to achieve fast and reliable autonomous robot deployments in order to tackle the problem that could not be solved in a different way.

The middleware embedded in networked robots should achieve the following tasks:

- interact with the robot firmware in order to drive the wheel motors and collect the information regarding the sensor output and the battery state,
- manage the communication with other robots and the base-station in the network,
- process both sensed information about the environment and the messages received from the neighbors in the network,
- react in a fast and reliable way to the events in the environment.

Although many robotic networks assume the existence and reliability of the global network infrastructure that can be used for the communication and information collection among the robots, this scenario cannot hold in the WSN applications. Therefore, the robotic network should rely on ad-hoc network infrastructure that allows any-to-any communication paradigm among robots.

The communication part of the robotic network still represents a challenge in practical implementations, the details of message exchange, information transmission and movement coordination still represents and actual problem although a significant amount of research has been conducted from the point of view of optimization theory. Furthermore, the dynamic nature to the robotic network imposes new problems regarding the perceived network neighborhood, energy consumption, environment exploration and collaborative path-planning. Finally, human aspect of the network control should be considered. Although automated, a robotic network should comprise a convenient manner of information capture and individual robot control, that can be controlled by human on demand. We try to give an answer to these challenges by proposing our networked robots system architecture.

3. RELATED WORK

In this section, we provide the state of the art works in the field of networked robotics and the robot middleware for deployment purposes. A robot middleware is defined as a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems [1]. It is a software layer that is positioned above the operating system and below the application layer. Parker proposes one of the first works that focuses on an architecture for mobile multi-robot management in [15]. In her thesis, Parker describes a software architecture that facilitates the cooperation between heterogeneous mobile robots. The proposed middleware allows the team of robots to perform a specific task and to respond to unexpected environmental changes. This paper is the basis of our work. However, the architecture in this paper includes more features. The most important are the communication aspects between the robots and the ease of implementing basic algorithm for wireless mobile sensor applications.

Further development of robotic middleware continued with Miro framework [19]. It comprises 3 layers: the device layer that provides interface abstractions for the hardware components and that is platform dependent, service layer that

provides services by using CORBA interface definition language, and the framework layer that provides the user with modules for robot control, localization, path planning, etc. In the same year, in [7] a hierarchical framework is proposed by the authors. This work mainly focuses on programming aspects and the formal definition related to the proposed high level language. Compared to our work, the proposed work only describes some basic functionality which does not include communication paradigms.

In [4], authors propose a control software to structure the multiple robots architectures. This work mainly focuses on high level primitive to create and provide a software interface to ease the programming. The proposed framework uses a Java virtual machine. The approach in our work does not rely on any specific software and use the robot low level primitives to build the robot cooperation and facilitating the programming methods by giving an access interface to each block. [3] describes a context-based design of robotics system. The goal of authors' approach is to improve the system performance, by using the features of the situation at hand. The work of Calisi et al. only focuses on the acquisition of context and its use. The same features are developed in this paper, but it also includes the communication part. In [11], the authors describe a multi-agent based solution to control and coordinate team-working mobile robots. The proposed work is very interesting and divided the architecture into three different blocks: physical, control and coordination. However, they do not describe in detail the communication modules which is one of the goals of our work.

One of the mostly used robotic middleware is Robotic Operating System [16]. It is a component-based platform supporting a client-server scheme for control flow and a publisher-subscriber scheme for data flow. ROS is actually not an operating system but rather a middleware that consists of nodes, messages, topics and services. Nodes communicate among each other by publishing messages and subscribing to published messages. However, its size and complexity makes it infeasible for mobile devices with constrained memory and computational power.

Other approaches to robotic middleware design and robotic frameworks such as Pyro [2], Claraty [12], Open RTM [14] and Orocos [18], can be found in survey papers of Mohamed et al. [10] and Sanfeliu et al. [17]. The interested reader can refer to [5] for a more detailed and recent survey on robotic middleware.

The middleware implementation proposed in this paper covers the basic aspects needed by the mobile robot deployment applications. Although there are some missing features that are present in other existing solutions, the middleware presented here is simple and portable, which makes it a good choice for small, energy and memory constrained robotic platforms used in swarm robotics.

4. ACRHITECTURE FOR MULTI-ROBOT COOPERATION

4.1 Middleware architecture

Middleware presented in this work is dedicated for the implementation on Wifibot mobile robots [20]. Wifibots are differentially driven, battery powered, mobile development platforms with integrated on-board computer (Figure 1). The base system comprises four wheel drive motor board controllable via RS232 link, 2 infrared range sensors, camera, mini-pci WIFI card, Intel Atom D510 Duo Core processor, operating system installed on 4G compact flash memory and a WIFI access point used for the tele-operation. Further details about the robot construction can be found on manufacturer's website[1]. Figure 2 presents the detailed schema of the proposed and implemented middleware architecture.

(a) (b)

Figure 1: Wifibot mobile robot (a) and a set of robots in an experimental scenario (b).

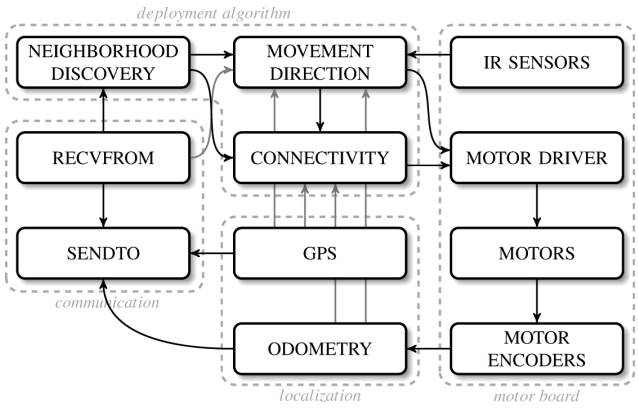


Figure 2: Robot middleware architecture.

The motor control block is the crucial block in the complete robot middleware architecture, since it is in charge of controlling the built-in sensors and motor drivers. When the movement decision has been brought in the deployment algorithm implementation block, or by the user command, it is processed by the motor driver and the motors are activated. Each motor is equipped with an encoder that gives the information regarding the quantity of the movement. This information is processed in the localization block. Two infrared range sensors are mounted on the front part of the chassis. In any case, the output from these two sensors is used to prevent the collision with obstacles during the deployment (including other robots as well). The output of these sensors overrides the movement commands given to the motor driver block, thus preventing any kind of collision. The battery block reports the electrical current and voltage to the user, and therefore can be used to evaluate the energy consumption of the robot during the deployment.

In this middleware implementation, two localization techniques are used. The first one is the absolute localization

[1]http://www.wifibot.com/

technique that relies on the global positioning system (GPS) receiver connected to the on-board computer. Without providing the details about the GPS localization technique, a general comment about the localization accuracy is that it is sufficient for the general use in automotive industry and that it can be sufficient for a variety of robot deployment applications. However, it is worth noting that all the GPS localization techniques need a signal from the satellites, which rules out this approach in confined and indoor environments. The second localization technique is the dead-reckoning localization based on the output from motor encoders. This kind of simple localization method is widely used since the position derived from the motor movements can be easily calculated in real-time. On the other hand, the biggest flaw of this kind of approach is the localization error that accumulates over time and the estimated location needs to be corrected in order to be usable. In the case of Wifibots, after the odometry parameter calibration, this method can be used for the indoor localization in the case where movement distances stay short. Another problem that arises with the use of dead-reckoning methods is the need for a relative coordinate system that needs to be shared among the set networked robots.

The communication block is in charge of information exchange with neighboring entities in the network – robots and base station. It is essentially composed of two parts: receiving and sending blocks. All the messages destined to a robot are processed by the receiving block, *Hello* messages are transfered towards the deployment algorithm block. *Control* messages destined to the robot that received the message processes the message and act accordingly to its contents by setting the parameters sent by the user. Robots in the network are used as relays, therefore *Data* and *Control* messages not destined to the current robot are transfered forward towards the sending block. The role of the sending block is twofold. First, it periodically checks the port where the localization block outputs robot's current position, then it creates the *Hello* message with the updated location information and broadcasts it. Second role is the *Data* and *Control* retransmission towards their destination. In case if there is a need for message transmission towards a destination other that actual robot, sending block consults the neighborhood table constructed by the neighborhood discovery block and sends the message towards its destination. This allows the multi-hop *Control* message transfer from the base station towards all the robots in the network, as well as information acquisition by relaying the *Data* messages from the robots towards the base station.It is worth noting here that all the messages that are passed between the blocks are also sent to a separate port on each robot, thus leaving space for other future upgrades and other block integrated in the system.

The first part of the deployment algorithm is the neighborhood discovery. Based on the information gathered upon the reception of *Hello* messages from neighboring robots, the neighborhood table is updated in the neighborhood discovery block. In a majority of geometry-based deployment techniques, a robot needs the exact position of the neighboring robots from the neighbor table, as well as its own precise position. All these calculations are covered by the direction calculation block. After the direction calculations, a robot has the defined direction and is ready to move. Although the movement has direction has been calculated and the robot is ready to move, if the obstacle is detected, all the movements (even the deliberate movement commands sent by the user) will be stopped. The second case is related to the network connectivity preservation and is processed in the connectivity block. After the movement direction has been calculated, the maximal allowed movement distance is calculated as well in order to prevent the network disconnections. After the maximal movement distance that keeps the network connected has been calculated, this information is sent towards the motor control block and transferred towards the motor driver. Depending on the set of commands and the deployment goal set by the user, the deployment algorithm block can serve different types of algorithms, from simple point-to-point linear movement to more complex deployment algorithm presented in Section 5.

4.2 Multi-robot network control

An integral part of the sensor deployment in most applications is the establishment of the data acquisition infrastructure. The dynamic nature of the robot deployment changes this paradigm in a sense that the network infrastructure must be auto-adaptable to environment conditions. Examples for this are the disaster areas where it is not possible to elect a set of sensors that will play the role of the communication backbone due to the possibility of sensor failures. The complete network should rather be equipped with the mechanisms of overcoming these types of unexpected environment behavior.

Setting up the network infrastructure may seem to be not a so complex task, however, the problems of cost and time to set it up can arise. The cost of the auto-adaptable network infrastructure becomes an obstacle in remote and large construction sites where the robotic network is used for a structure and machinery health monitoring. In most military applications that require fast and reliable response to environmental changes, the network infrastructure reaction time represents one of the major issues.

Our approach relies on the use of connectionless UDP (User Datagram Protocol), that allows us to boost the robustness of the network. Prior to actual information routing in the network it is necessary to construct a routing scheme. The base station broadcasts the gradient control message and the message is flooded in the network in the multi-hop manner. During this process, on each message reception, a robot increments the gradient value in the message or discards the message if it is already received. After all the robots have received the message from the base station, the process of establishing the robot gradient is completed. When a robot has some information to transmit towards the base station, it sends it to its neighboring robot with lowest gradient. Due to the network dynamics, robots can lower or increase their gradient numbers in case of changes in the neighborhood.

In order to allow human interaction and the control of individual robots, we introduced the *Control* messages in the network. In this way, a user can use the base station control application to control a set or an individual robot in the network (Figure 3). *Control* messages are transmitted in a multi-hop fashion towards the increasing gradient robots.

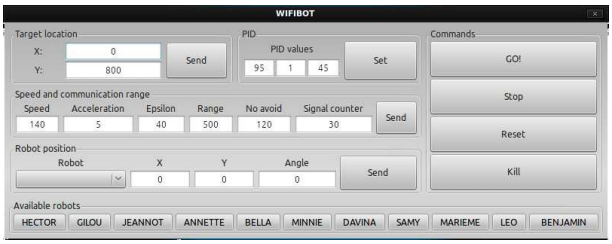


Figure 3: Multi-robot control application.

5. APPLICATIONS IN WSN

Robotic networks in the context of WSN suffer from the same problem that strikes any product in development – the compromise between thorough testing and the necessity to move a designed system to the market quickly. Full system testing is impossible to achieve, above all in the design of mobile robots dedicated for the aforementioned applications, since it is impossible to envisage all the possible situations and hazards that could appear in the real world. In this section, we use 3 examples of applications, in order to show that our networked architecture can be used both for real deployment and for network testing.

5.1 PoI coverage

Covering Points of Interest (PoI) in the deployment field is one of the standard applications of WSN. More information about the PoI coverage with mobile sensors can be found in [6].

In order to ease the understanding of the deployment procedure, we use the following definitions and notations for the network model. Let $G(V, E)$ be the graph representing the sensor network composed of mobile robots. V is the set of vertices each one representing a robot and $E \subseteq V^2$ is the set of edges, so that $E = \{(u, v) \in V^2 \mid u \neq v \ \wedge \ d(u, v) \leq R\}$, where $d(u, v)$ is the euclidean distance between robots u and v and R is the communication range. The value of the R can be set manually by using the base station application (Figure 3). $N(u)$ is the set of 1-hop neighbors of robot u, so that $N(u) = \{v \in E \mid d(u, v) \leq R\}$. We assume that the positions of robot u and PoI p are denoted by $u(x, y)$ and $p(x, y)$, respectively. We assume that at the beginning of the deployment network is connected and the robots are randomly spread out around the base station.

In order to cover the PoI, robots move toward one predefined point that could be the PoI itself or the barycenter of PoIs. While robots are moving, they must maintain connectivity with their Relative Neighborhood Graph (RNG) neighbors. Following the rules of connectivity preservation, even if a robot does not cover the PoI, it must stop moving in order to maintain the connection. The direction of a robot is given by the following unit vector : $\overrightarrow{\Delta} = \overrightarrow{d_p}/||\overrightarrow{d_p}||$, where $\overrightarrow{d_p}$ is the vector connecting the current position of the robot with the PoI (Figure 4). When robot has computed $\overrightarrow{\Delta}$, it will move towards this direction. In the case of the RNG neighbor constraints, the movement vector of a robot is $\overrightarrow{m} = d \cdot \overrightarrow{\Delta}$, where d is the maximum distance that the robot can travel while maintaining connectivity. We set the following condition for the maximum distance d: $d \leq (R - d^+(u))/2$, where R is the communication range, $d^+(u)$ is the distance from robot u to its farthest RNG neighbor. This condition ensures that robot u and $RNG^+(u)$ remain connected to each other, even in the case of movements in opposite directions.

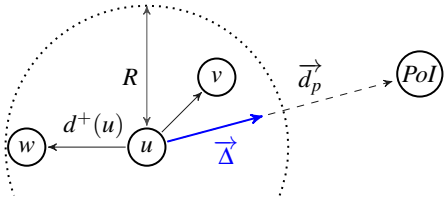


Figure 4: PoI coverage algorithm illustration.

In the `direction` block, the robot u computes its direction $\overrightarrow{\Delta}$ based on its own position and the coordinates of the PoI. In the `connectivity` block, the robot u calculates the distance to travel and performs the actual movement. Connectivity constraint ensures that if $v = RNG^+(u)$, then $d(u, v) \leq d^+(v)$. This inequality ensures that the connectivity is still preserved between sensors u and v. If a robot detects an obstacle during the deployment, it tries to cover the auxiliary PoI ($p_A(x_A, y_A)$) based on the gathered local information about the obstacle. After the auxiliary PoI is reached, it continues with initial PoI coverage steps. After the movement is done, the steps are repeated until the PoI is covered (WiDep, Algorithm 1).

Algorithm 1: Wifibot deployment algorithm (WiDep).

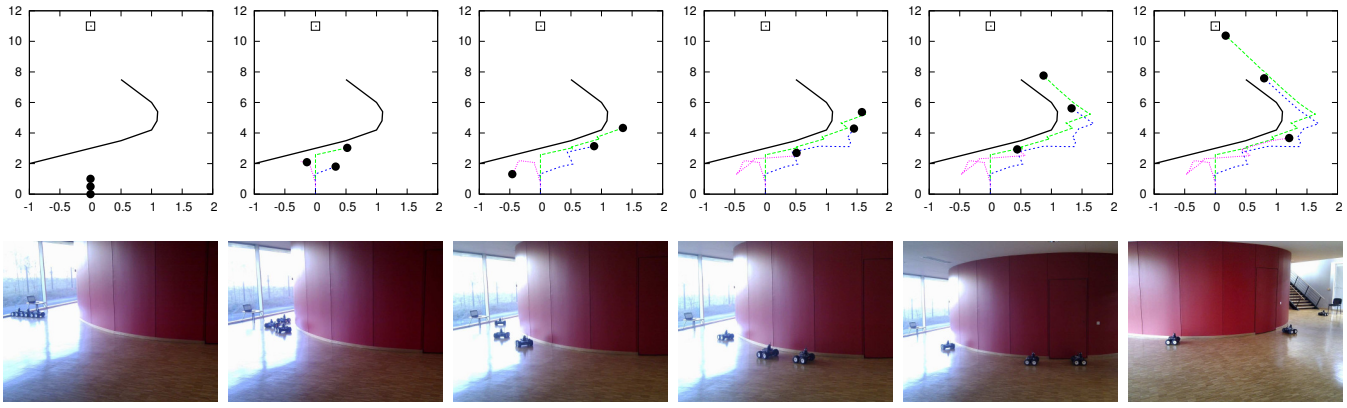
Require: The PoI location $p(x, y)$.
Ensure: The coverage of the PoI $p(x, y)$.
1: **repeat**
2: $\overrightarrow{\Delta} = \dfrac{\overrightarrow{d_p}}{||\overrightarrow{d_p}||}$; $d = (R - d^+(u))/2$
3: Movement using the direction $\overrightarrow{\Delta}$ and distance d.
4: **if** obstacle detected **then**
5: Run WiDep for $p_A(x_A, y_A)$
6: **end if**
7: **until** the PoI is reached

In this example application, the Wifibot's on-board computer serves us as the platform for the deployment algorithm implementation, while the motor driver provides us with the real-time information on power drawn from the battery. Figure 5 shows the example of deployment with obstacles, where 3 Wifibots are used. In order to cover the PoI $p(0, 11)$, the communication range is set to $4.5m$. This example illustrates the behavior of the implemented middleware architecture dedicated for multi-robot deployment.

5.2 Multi-hop video transmission

Next example describe the use of middleware for the multi-hop video transmission. We considers the behaviour of two robots (R_1 and R_2) that are used as mobile transmission relays between the video transmission source S and destination D nodes (Figure 6) and its impact on video transmission quality. The robots can move on the straight line between source and destination in order to ensure good quality to the transmitted video, by measuring the transport layer properties of the link – ping delay and data loss.

Our goal in these experiments is to examine if it is possible to use a simple movement algorithm to achieve better video



(a) 0s (b) 10s (c) 30s (d) 40s (e) 50s (f) 60s

Figure 5: Wifibot deployment with an obstacle

transmission quality and does it affect the parameters of the transmitted video. Our assumption is that the mobile robots are going to be placed at the distance $d = 2(n_{node} - 1)$ from the source node (where n_{node} represents the robot's position in the network and the distance is given in meters). In the context of this paper, the further details about the positioning algorithm are not necessary.

It is worth noting that all the experiments are done indoors and therefore the maximal distance between the source and the destination node is limited by the free space withing the testing area (30 meters in our case). Furthermore, the localization technique in this case relies only on the wheel odometry.

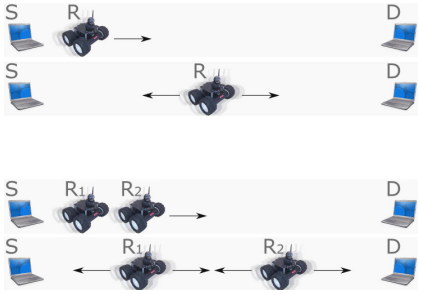


Figure 6: Two video transmission scenarios with mobile robots.

After the video transmission being evaluated, we present the output of the positioning algorithm gathered by the inner sensors. Two cases are presented, with one and two robots autonomously positioning themselves in between source and destination nodes. In the Figures 7 and 8, the time units are represented by the movement decision steps from the positioning algorithm, where each phase lasts for $16s$.

The evolution of the position of the single mobile robots between static source and destination nodes is depicted in Figure 7. We can see that the robot positions itself in the low data loss region after approximately 10 movement decisions, and it slightly oscillates around the middle point until the end of our test. This is an expected behavior since we saw that the lowest loss region of the video transmission link is exactly the middle region regarding the distance. Two mobile robot scenario shows satisfactory results regarding the robot placement during the algorithm run (Figure 8). As it could be expected, robots choose approximately equidistant positions to preserve low data loss.

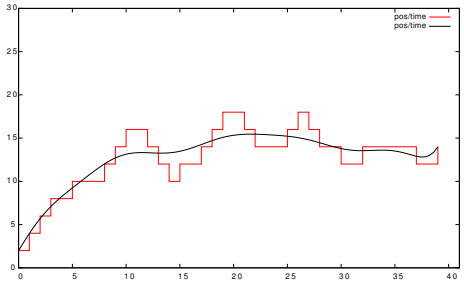


Figure 7: Trajectory of the robot running the positioning algorithm while transmitting the video.

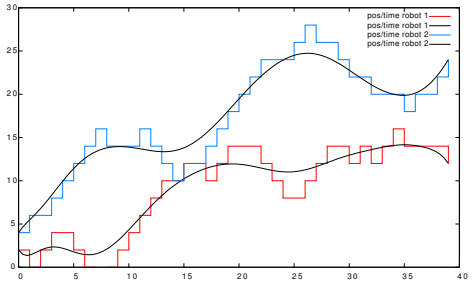


Figure 8: Trajectories of two robots running the positioning algorithm.

This example shows that our middleware architecture is usable in the context of WSN where there is a need for video signal acquisition, while performing the movements in order to improve the transmitted video quality.

5.3 Energy consumption evaluation

In this final example, we use the proposed middleware in order to analyze the Wifibot power consumption. In all

the deployment scenarios, when the movement is being performed, the robot varies its movement velocity from the zero value v_0 (robot does not move, *standby mode*) to the desired speed $v < v_{max}$ (*movement mode*, where v_{max} represents the maximal allowed robot velocity, configurable in base station application). Figure 9 shows the evolution of desired and actual speed of Wifibot during the movement of $5m$. These measures are obtained from the output of the inner sensors.

The speed in Figure 9 is represented in motor encoder ticks per $50ms$, where the value of 140 corresponds to the velocity of $0.5m/s$. Furthermore, the same figure shows the real-time consumed power during the same movement. We use the mean power value to overcome the noisy measurements. It is worth noticing the impact of motor speed controller on the actual obtained movement velocity and power, that is captured by the sensory block.

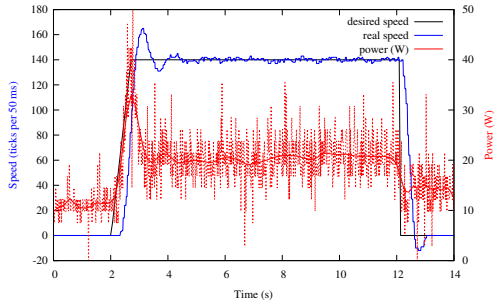


Figure 9: Speed and power data during the $5m$ movement. The speed of 140 ticks per 50 ms corresponds to the speed of $0.5m/s$.

In order to quantify the robot energy consumption while not moving (standby mode), we measure the average power during the period of $T = 600s$. Obtained results from the inner sensors show that the power in standby mode is $P_0 = 8.568$ W.

Likewise, in order to quantify the power drawn from the battery during the movement, we vary the desired speed from $0.1m/s$ to $0.9m/s$ with the step of 0.1 and measure the power during the period of $T = 120s$. The obtained results are presented in Figure 10 and the power depending on the movement velocity, can be expressed as: $P_{mov}(v,t) = 11.55v(t) + 14.838$W The robot varies its velocity during the deployment, thus both the velocity $v(t)$ and the power $P_{mov}(v,t)$ are the functions of time.

The energy consumption E during the period T is calculated as the time integral of movement power. Therefore, by combining P_0 and $P_{mov}(v,t)$ into the expression for the consumed energy E, we get the desired energy model:

$$E = \begin{cases} \int_0^T P_0 \, \mathrm{d}t = 8.568T[J] & \text{if } v = 0 \\ \int_0^T P_{mov}(v,t) \, \mathrm{d}t = 11.55 \int_0^T v(t) \, \mathrm{d}t + 14.838T[J] & \text{if } v > 0. \end{cases}$$

The energy consumption simulated for each sensor and the energy consumed during the implementation is presented in Figure 11. Measured results show that the energy consumption values are highly correlated, with a small difference in the energy of the robots closer to the base station

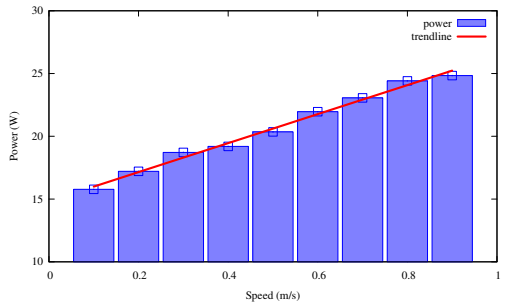


Figure 10: Average power during the movement with different speeds (from 0.1 to $0.9m/s$) obtained as the output of the inner sensors.

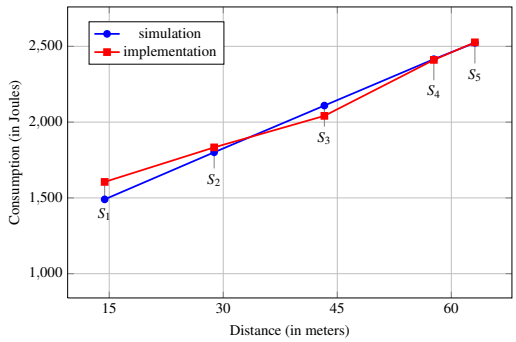


Figure 11: Simulation and implementation results for total consumption with relation to the robot position.

and high accuracy of predicted energy consumption for the robots that are closer to the PoI.

Obtained results show that the proposed middleware architecture allows the user to monitor the energy consumption on demand by the use of the control application.

6. CONCLUSIONS AND FUTURE WORK

In this work, we described the robot middleware architecture that allows networked multi-robot control and data acquisition in the context of wireless sensor networks. We presented three specific examples of robot network deployment and illustrated the proposed architecture usability: the robotic network deployment with the goal of covering the Point of Interest, adaptable multi-hop video transmission scenario, and the case of obtaining the energy consumption during the deployment. Obtained experimental results show that our middleware and networked robot communication architecture is feasible in the context of WSN with mobile robots.

Many different challenges arise in the field of networked robots for the applications of wireless sensor networks. One of the problems is the problem of scalability in the network, notably regarding the control and information acquisition among a large number of robots. New data routing protocols should be designed in order to cope with the unpredictability and dynamic nature of the robotic networks. Another fundamental problem is the problem of *security*. It is related both to the transmitted information security as well as the robot and sensory devices. Networked robots security

issues are closely related to the self-reconfiguration abilities, followed by the detection of the errors in the communication processes. Our future work will focus on these issues.

Acknowledgements

This work was partially supported by a grant from CPER Nord-Pas-de-Calais/FEDER Campus Intelligence Ambiante and the French National Research Agency under the grant number: ANR VERSO RESCUE (ANR-10-VERS-003).

Bibliography

[1] D. Bakken. Middleware. *Encyclopedia of Distributed Computing*, 2001.

[2] D. Blank, D. Kumar, and B. M. College. Pyro: A python-based versatile programming environment for teaching robotics. *ACM Journal on Educational Resources in Computing*, 3(4):1–15, 2003.

[3] D. Calisi, L. Iocchi, D. Nardi, C. M. Scalzo, and V. A. Ziparo. Context-based design of robotic systems. *Robotics and Autonomous Systems*, 56(11):992 – 1003, 2008.

[4] L. Cragg, H. Hu, and N. Voelker. Modularity and mobility of distributed control software for networked mobile robots. In D. Brugali, editor, *Software Engineering for Experimental Robotics*, volume 30 of *Springer Tracts in Advanced Robotics*, pages 459–484. Springer Berlin Heidelberg, 2007.

[5] A. Elkady and T. Sobh. Robotics middleware: A comprehensive literature survey and attribute-based bibliography. *Journal of Robotics*, 2012.

[6] M. Erdelj, T. Razafindralambo, and D. Simplot-Ryl. Covering points of interest with mobile sensors. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):32–43, 2013.

[7] R. B. Fierro, A. K. Das, J. R. Spletzer, J. M. Esposito, V. Kumar, J. P. Ostrowski, G. J. Pappas, C. J. Taylor, Y. Hur, R. Alur, I. Lee, G. Z. Grudic, and B. Southall. A framework and architecture for multi-robot coordination. *I. J. Robotic Res.*, 21(10-11):977–998, 2002.

[8] G. Kantor, S. Singh, R. A. Peterson, D. Rus, A. K. Das, V. Kumar, G. A. S. Pereira, and J. R. Spletzer. Distributed Search and Rescue with Robot and Sensor Teams. In *FSR*, pages 529–538, 2003.

[9] T. Lochmatter, X. Raemy, and A. Martinoli. Odor source localization with mobile robots. *Bulletin of the Swiss Society for Automatic Control*, 46:11–14, 2007.

[10] N. Mohamed and J. Al-Jaroodi. Characteristics of middleware for networked collaborative robots. In *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, pages 524–531, 2008.

[11] M. Mouad, L. Adouane, P. Schmitt, D. Khadraoui, and P. Martinet. Control Architecture for Cooperative Mobile Robots using Multi-Agent based Coordination Approach. In *6th National Conference on Control Architectures of Robots*, page 15 p., Grenoble, France, 2011. INRIA Grenoble Rhône-Alpes.

[12] I. A. D. Nesnas, R. Simmons, and D. Gaines. Claraty: Challenges and steps toward reusable robotic software. *International Journal of Advanced Robotic Systems*, 3(1):23–30, 2006.

[13] C. Nikolaus and A. Martinoli. Multirobot Inspection of Industrial Machinery From Distributed Coverage Algorithms to Experiments with Miniature Robotic Swarms. *IEEE Robotics & Automation Magazine*, 16:103–112, 2009.

[14] K. Ohara, T. Suzuki, B. K. N. Ando, K. Ohba, and K. Tanie. Distributed control of robot functions using rt middleware. *Proceedings of the International Joint Conference (SICE-ICASE âĂŹ06)*, pages 2629–2632, 2006.

[15] L. Parker. Alliance: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In *In Proc. IROS 1994*, pages 776–783, 1994.

[16] ROS. Robot operating system, www.ros.org.

[17] A. Sanfeliu, N. Hagita, and A. Saffiotti. Network robot systems. *Robotics and Autonomous Systems*, 56(10):793–797, 2008.

[18] P. Soetens. Rtt: Real-time toolkit, 2010.

[19] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar. Miro – middleware for mobile robot applications. *IEEE Transactions on Robotics and Automation*, 18(4):493–497, 2002.

[20] Wifibot. Mobile robots, www.wifibot.com.